IN THE CLAIMS:

Please amend claims 12 and 14, respectively.

1. (Original) A method, comprising:

encountering a function call instruction that calls a called function during program execution;

saving a return address in a first stack and in a second stack, the return address containing an instruction to be executed after execution of the called function;

executing the called function; and

determining if the return address stored in the first stack matches the return address stored in the second stack.

2. (Original) The method of claim 1, further comprising generating an exception if the return addresses do not match.

3. (Original) The method of claim 2, further comprising executing exception handling code if an exception was generated.

4. (Original) The method of claim 3, wherein the exception handling code determines what value to pass to a program pointer based on the return addresses retrieved from the first and second stack.

5. (Original) The method of claim 3, wherein the exception handling code terminates execution of the program.

6. (Original) A method, comprising:

processing instructions within a virtual machine;

saving a return address in a first stack and in a second stack, the return address being an address at which program execution is to resume after execution of a called function;

comparing the return addresses saved in the first and second stack upon execution of the called function; and

exiting the virtual machine if the return addresses do not match.

7. (Original) The method of claim 6, further comprising passing control to an exception handler.

8. (Original) The method of claim 7, wherein the exception handler determines if the return address from the first stack or the return address from the second stack is to be used as a value for an instruction pointer.

9. (Original) A method, comprising:

creating first and second stacks for a program during execution of the program;

encountering a function call to a called function;

storing data for the called function and a return address in the first stack;

storing the return address in the second stack; and

passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function.

10. (Original) The method of claim 9, wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer.

11. (Original) A processor, comprising:

memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction calls to a called function is encountered during program execution;

function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations, the return address being an address at which program flow is to resume after execution of the called function; and

buffer overflow control logic to determine if the return address retrieved from the first stack matches the return address retrieved from the second stack, upon execution of the called function.

12. (Currently Amended) The processor of claim 11, wherein the function call~~ed~~ logic and the buffer overflow control logic comprises microcode stored within the processor.

13. (Original) A system, comprising:

a memory; and

a processor coupled to the memory, the processor comprising memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction that calls a called function is encountered during program execution;

function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations, the return address being an address at which program flow is to resume after execution of the called function; and

buffer overflow control logic to determine if the return address retrieved from the first stack matches the return address retrieved from the second stack, upon execution of the called function.

14. (Currently Amended) The system of claim [[11]]13, wherein the memory management logic, the function call logic ~~command~~, and the buffer overflow control logic comprise microcode stored within the processor.

15. (Original) A computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

encountering a function call instruction that calls a called function during program execution;

saving a return address in a first stack and in a second stack, the return address containing an instruction to be executed after execution of the called function;

executing the called function; and

determining if the return address stored in the first stack matches the return address stored in the second stack.

16. (Original) The computer readable medium of claim 15, wherein the method further comprises generating an exception if the return addresses do not match.

17. (Original) A computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

processing instructions within a virtual machine;

saving a return address in a first stack and in a second stack, the return address being an address at which program execution is to resume after execution of a called function;

comparing the return addresses saved in the first and second stack upon execution of the called function; and

exiting the virtual machine if the return addresses do not match.

18. (Original) The computer readable medium of claim 17, wherein the method further comprises passing control to an exception handler.

19. (Original) A computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

creating first and second stacks for a program during execution of the program;

encountering a function call to a called function;

storing data for the called function and a return address in the first stack;

storing the return address in the second stack; and

passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function.

20. (Original) The computer readable medium of claim 19, wherein the exception handler determines if the return address from the first stack and the return address from the second stack is to be used as a value for an instruction pointer.